

**What Is Claimed Is:**

- 1           1.       A method to efficiently realize class initialization barriers in a  
2       multitasking virtual machine, wherein class loading always takes place before  
3       class initialization, and wherein a class initialization barrier guarantees that a class  
4       is initialized before the class is first used by a program, comprising:  
5                associating a shared runtime representation of the class with a task class  
6       mirror table that comprises at least one entry per-task, including an initialized  
7       entry, for a plurality of tasks, wherein each entry holds either a null pointer value  
8       or a non-null pointer to a task class mirror object, wherein all entries of a task  
9       mirror table that hold a non-null pointer value and that are associated with a same  
10      task hold a pointer to a same task class mirror object, wherein the task class mirror  
11      object holds a task private representation of the class for that task;  
12               using the initialized entry of a task in the task class mirror table to  
13      determine whether this task has initialized the class associated with the task class  
14      mirror table; and  
15               accessing the task class mirror object associated to a particular task.
- 1           2.       The method of claim 1,  
2               wherein each task is associated with a unique integer value;  
3               wherein the unique integer value is used to compute a byte-offset from a  
4       beginning of task class mirror tables that can be used to retrieve from the  
5       initialized entry of any task class mirror table the pointer to the task class mirror  
6       object; and  
7               wherein a computed byte-offset to the initialized entry is stored in a  
8       descriptor of a plurality of threads executing on behalf of a corresponding task.

1           3.       The method of claim 2, further comprising:  
2           creating the task class mirror table and associating the task class mirror  
3 table with the shared runtime representation of the class upon creation of the  
4 shared runtime representation of the class; and  
5           setting all entries of the task class mirror table to the null pointer value.

1           4.       The method of claim 3, further comprising:  
2           examining the initialized entry of the task in the task class mirror table  
3 associated with the class in order to determine if that task has initialized the class,  
4 wherein the byte-offset to the initialized entry from the beginning of the task class  
5 mirror table is obtained from the descriptor of a thread performing an examination  
6 on behalf of the task; and  
7           initializing the class by the task if the class is not already initialized,  
8 wherein a null pointer stored at the initialized entry indicates that the class has not  
9 initialized the task.

1           5.       The method of claim 4, further comprising:  
2           upon completion of initialization of the class by the task, setting the  
3 initialized entry of the task class mirror table associated with the class to the task  
4 class mirror object that holds a representation of the class that is private to the  
5 task; and  
6           setting this task class mirror object to a fully initialized state.

1           6.       The method of claim 5, wherein task class mirror tables associated  
2 with classes that have a non-empty initialization function includes one resolved  
3 entry per-task in addition to one initialized entry per-task, for the plurality of  
4 tasks.

1           7.       The method of claim 6, wherein task class mirror tables associated  
2 with classes that have an empty initialization function include one resolved entry  
3 per-task in addition to an initialized entry per-task, for the plurality of tasks.

1           8.       The method of claim 7, further comprising:  
2           upon loading any class by the task, creating the task class mirror object  
3 that holds the task private representation of the class;  
4           setting the task class mirror object's state to loaded; and  
5           assigning the task class mirror object's pointer to a resolved entry of the  
6 task class mirror table associated with the class for that task.

1           9.       The method of claim 8,  
2           wherein the task class mirror table is arranged so that the resolved entry  
3 and the initialized entry for the task are consecutive; and  
4           wherein the byte-offset to the resolved entry can be computed from the  
5 byte-offset to the initialized entry for a same task by adding a size, expressed in  
6 number of bytes, of the pointer to the task class mirror object.

1           10.      The method of claim 8,  
2           wherein the task class mirror table is arranged so that the resolved entry  
3 and the initialized entry for the task are separated by half of a total number of  
4 entries in the task class mirror table; and  
5           wherein the byte-offset to the resolved entry can be computed from the  
6 byte-offset to the initialized entry for a same task by adding a size, expressed in  
7 number of bytes, of half the total number of entries in the task class mirror table.